EECS 448

Team 16

Code Review

Our team decided to follow the Inspection process when it came to the code review. The goal of this is beyond constructing questions and facilitating discussions of our functions used to create the game. This process includes an overview, preparation, inspection, a rework, and a follow-up. Each time we noticed a defect or error in our game, we would relay the issue into our group chat. There we would discuss the problem in detail to fully understand why this occurred. Then one member would take it upon themselves to handle the issue and document how they resolved it.

As we began to add our new features from the skeleton presented in Project 3, our first error appeared. On occasion, the moving ball would pass through the paddle and hit the bottom of the screen. The frequency of this event increased as the speed of the ball increased in the game. The two events were not mutually exclusive. Kaitlyn was assigned to fix this fault. She approached this defect by altering the hitpaddle() parameter from y == 550 to ( y>= 550 && y <= 552). Since this method is called on 5ms intervals, this opens a path for the ball to pass through the paddle without being detected. Therefore, expanding on the range will ensure that the ball will be detected and bounce off the paddle.

The second fault we detected involved the screen and the buttons. When you pressed start to play the game, the screen appeared blank on the users end and none of the buttons when clicked executed its function properly. Cooper was assigned to fix this fault. He resolved this issue by taking the time to go through the source code and add semi-colons that had been previously missed in many functions. He removed an if statement held in the score function since it was already mentioned in the hit detection function. The if statement declared that if the score was less than zero then ignore the hits on a brick. However, the score was initialized at zero as a global variable.

Adding new and more difficult levels to the game caused level 2 to break. Within this level, the bricks did not disappear after contacting the ball. Josh was assigned to fix this fault. His solution involved interacting with the health of the level 2 bricks. In this area, the bricks were not decreasing after each contact to the ball, so Josh subtracted one from the health of the bricks that were hit. This ensured that all levels and intensity of the game would function effortlessly.

After this level 2 affair was fixed, this led us to discover a new bug in the software. When the ball would touch the floor, it didn’t recall its past lives. The ball would be positioned back above the paddle where you would have to click to shoot again. The bricks and score displayed on the screen would not consider any of the hits that user made previously. Tristan was assigned to fix this fault. He approached this task by creating a function to check the number of lives in the game and to make sure the hearts displayed at the top of the screen only display the number of lives left. More so, he created another function to reset the position and direction of the ball when you lose a life.

We discovered a big fault towards the middle of our project where the count of the scoreboard was off. The board was only changing when a brick disappeared and did not account for each time the ball hit a brick (since there are levels that require the ball to hit a brick more than once before disappearing). Cooper was assigned to fix this fault. He did so by changing the total score for the current level to the entire score of all the brick health. He then set this function to when the score is equal to the maximum possible score. Finally, he switched it to the next level to complete the defect.